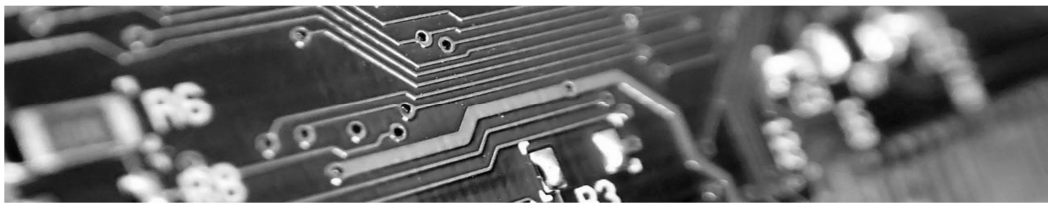
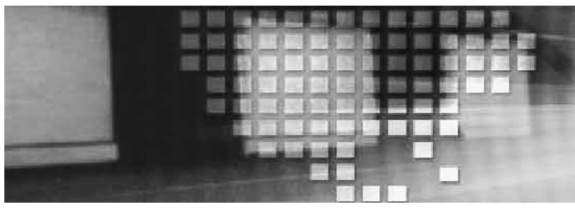
SARATOGA DATA SYSTEMS





Flume User Manual
Version 2.3

Software License Notice

Your license agreement with Saratoga Data Systems, Inc. specifies the permitted and prohibited uses of the product. Any unauthorized duplication or use of Flume in whole or in part, in print, or in any other storage and retrieval system is forbidden except licensed users may copy this manual for internal use only. Saratoga assumes no responsibility for the use or reliability of software on equipment that the license agreement does not describe.

Disclaimer

Saratoga Data Systems, Inc. software and documentation has been tested and reviewed. Nevertheless, Saratoga Data Systems, Inc. makes no warranty or representation, either express or implied, with respect to the software or documentation included. In no event will Saratoga Data Systems, Inc. be liable for direct, indirect, special, incidental, or consequential damages resulting from any defect in the software or documentation included with these products. In particular, Saratoga Data Systems, Inc. shall have no liability for any programs or data used with these products, including the cost of recovering such programs or data.

Copyright Notices

Flume incorporates a number of free or open source software packages. Notices required by the developers are in Appendix D: Open Source Copyright Notices.

Patent and Trademark Notice

The Flume Network Optimization™ protocol is protected by patent 8,310,920, with additional patents pending. Flume Network Optimization™ is a registered trademark of Saratoga Data Systems, Inc.

Table of Contents

1. Introduction

1.1 The Flume Protocol and Software

Flume Network Optimization™ is a patented protocol using UDP plus TCP for 100% reliable copying of files over networks with high latency, errors, and/or intermittency. By making better use of the available network bandwidth, the software application ***Flume*** can often send files much faster than with pure TCP.

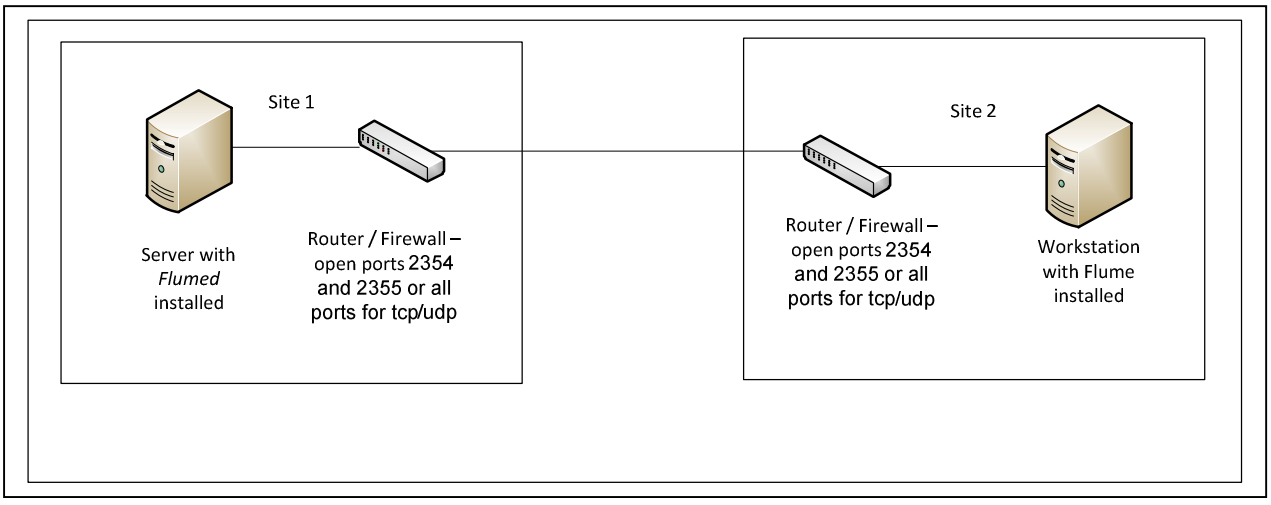
Flume is a client-server software solution which transfers files over a network. The ***Flume*** client program copies files to or from a remote computer that is configured with the ***Flume*** server and helper programs.

Flume does not require any special network hardware or any modifications to the host operating system.

***Flume* 2.3** consists of

1. ***flume*** - a single executable program file - the client, with several other names hard linked to the same executable:
2. ***flumed*** - the server
3. ***flumec*** - an interactive configuration file builder
4. ***flumep*** - the proxy server
5. ***flume_sig*, *flume_delta***, and ***flume_merge*** - three helper programs for computing and using file differences. These analyze and use the differences between updated source files and original older versions at the destination in order to only send changes.

1.2 Flume Setup Diagram



1.3 Support

Visit our web-site, www.saratogadata.com, for additional documentation and application notes.

Send your questions, comments, and problems regarding ***Flume*** to flume@saratogadata.com. You may also call +1(408)824-5321 with support questions.

Send your request for ***Flume*** licenses to sales@saratogadata.com.

2. Requirements

2.1 Flume hardware requirements

Flume is a 32-bit x86 program that runs on 32-bit x86 or 64-bit x86_64 Intel or AMD processors.

Almost any system from the last five to seven years will have enough power to run ***Flume***. Successful tests have been run on low end hardware such as a 900 MHz AMD Duron with 256 MB RAM and Athlon 64 systems in the 2 GHz clock range with single core processors.

Virtual Machines

Flume can also run in a virtual machine. Our public.saratogadata.com test server in Santa Clara, California, is a virtual machine running CentOS 5 with 512 MB virtual RAM and two 8 GB virtual disks. This is hosted on a quad-core Xeon system running 64-bit CentOS 5 with 8 GB RAM and two 500 GB disks running VMware Server 1.0.5. With a 100 TX full-duplex Ethernet connection to the Internet we detect no significant performance degradation due to the virtualization. With this system, we get over 40 M bps Flume transfers (vs. 4 Mbps with FTP) to China using the public Internet.

CPU load and usage

Due to the aggressive nature of the protocol, ***Flume*** will typically use 99%+ of one processor when transmitting data. A multi-core system (or dual-core virtual system) mitigates this issue.

The receiving end of ***Flume*** will use much less CPU depending on the total rate of the data coming in. Five to 10% usage of a single CPU is normal.

2.2 Flume operating system requirements

Flume consists of user-mode programs that require no modifications to the operating system or any special hardware. These programs do not require root privileges. It is possible to install Flume for operation entirely as a non-root user.

We support only the Linux 2.6 kernel for Flume but we support several different Linux distributions:

1. **32-bit Linux 2.6** kernel on Red Hat Enterprise Linux (**RHEL**) or RHEL clone: **CentOS 4, 5,** and 6, Ubuntu 7.10 and 10, and SUSE Linux Enterprise Server (**SLES**) **10**

2. **64-bit Linux 2.6** kernel on 64-bit x86_64 Red Hat Enterprise Linux (**RHEL**) or RHEL clone: **CentOS 4,** 5, and 6 and SUSE Linux Enterprise Server (**SLES**) **10**

At this time, the regular Linux 2.6 kernel release of ***Flume*** does not work on SUSE Pro 9 or SUSE Linux Enterprise Server (SLES) 9 but a modified version is available that works with SLES 9 with either 32-bit and 64-bit kernels.

Solaris is NOT supported in any version or hardware platform.

2.3 Additional Software/ Configuration Needed / Recommended

There are several packages that should also be available and configured on a machine running ***Flume***.

1. On 64-bit Linux, support for 32-bit code and libraries must be installed (and are not in some distributions): glibc.i686, libstdc++.i686, pam.i686 On CentOS / RHEL this can be done with...
    a. Update the 64-bit libraries first with:
       sudo yum update glibc libstdc++ pam
    b. Install / update the 32-bit libraries with
       sudo yum install glibc.i686 libstdc++.i686 pam.i686
2. A **text editor** such as ***vi*** or ***emacs*** is recommended to edit ***Flume*** configuration files.
3. **ssh** is the preferred mechanism for the ***Flume*** client to summon the ***Flume*** server. Therefore as a minimum the client system needs the ssh client and the server needs the ssh server.
4. **ssh** login without a password prompt should be configured through the use of a public/private key pair.
5. Any **firewall** such as the standard ***iptables*** package should be either disabled or set to pass Flume ports.
6. Meaningful host **names** and the ability to address systems by name will reduce confusion both at evaluation time and in later inspection of console, client, and server logs.

3. Installation

3.1 General

Flume is delivered as a *gzip* compressed tar file plus a ZIP file with a copy of the documentation and an installation script (plus an additional script to clean an older version of ***Flume*** from your system if necessary).

As of the 2.3.3 release ***Flume*** is also delivered as an RPM file for easier installation and un-installation using the Red Hat Package Manager.

Flume is most easily installed using the *root* login on your systems. In this manner it may be configured for use by other users. The supplied configuration files and installation script are for a standard installation done as root. However, Flume can be installed, configured, and run without root access. Please contact flume@saratogadata.com for instructions on this or a preconfigured package designed to install in ~/flume of a regular user.

In addition to ***Flume*** installation files you need a Flume 2.3 license file. ***Flume*** will not run without a valid license file. You can get a license file from Saratoga Data Systems, Inc.

NOTE: Other versions of ***Flume*** have different license files. Be sure you have a Flume 2.3 license.

3.2 Installation

Get all necessary files from Saratoga Data Systems, Inc.

- ftp://public.saratogadata.com/pub/sw/flume_2.3/
- Flume 2.3 User Manual
- Flume 2.3 QuickStart Guide
- *README.txt
- .zip archive with installation script and additional documentation
- .tar.gz with ***Flume*** software.
- .sha1sum checksum files for the ZIP and .tar.gz archives.
- flume*.rpm file for RPM based installation

Installation and configuration – on each end

- *Run as root for the standard installation.*
- Copy all the files to a convenient directory
- New style installation using .rpm (PREFERRED)
  - **rpm -Uvh flume*.rpm**
  - Copy your license file as supplied by email to /etc/flume/config.
- Old style installation with .tar.gz and .zip

- Copy your license file as supplied by email to the same location.
- Unzip the .zip archive
- Make the install_flume.bash script executable:
  chmod -v +x *.bash
- Run the installation script:
  ./install_flume.bash

o Configure flume for ***your*** network

- Ensure all files and directories are in locations accessible to you
- CRITICAL: Set initial value for ***speedMaximum*** parameter based on current ftp or other TCP transfer rate but larger values must be used once a successful run has been done at the initial setting.

3.3 Installing without root privileges

The file */etc/flume/config* specifies all ***Flume*** file locations other than itself. The location of this configuration file for all flume programs can be specified using the **–f** <fullPathToFile> parameter. Alternatively, for a non-root installation a symbolic link can be used for /etc/flume/config to a file that is editable by you. This will probably need to be created by someone who does have root access.

The other element of running with an alternative configuration file is that ***flumed*** and any other ***Flume*** helper programs must be invoked with
`-f /path/to/alternative/config.`

This is easily done with simple shell scripts either in ~/bin for the user or /usr/local/bin. For example:

```
[flume@b64-centos4x32 ~]$ cat bin/flumed
#!/bin/bash
~/flume/bin/flumed -f ~/flume/etc/flume/config $*
```

If you need help with this or would like a preconfigured Flume installation for installation and use by a non-privileged user please contact Saratoga Data Systems, Inc. at flume@saratogadata.com.

3.4 Installing the Flume License File

Flume requires a FLEXlm license file to run at each end of the connection. ***Flume*** will not run unless you have a valid and properly installed license file. You may request a license file from Saratoga Data Systems, Inc. by sending e-mail to flume@saratogadata.com.

NOTE: The current version of Flume uses a different license than earlier versions. You cannot use an old license.

Local license files

The ***Flume*** license file (s) can be installed anywhere in the file system. Set the *flexlmLicenseDirectory* configuration parameter to the name of the directory containing the license file. This must be done on each machine on which ***Flume*** runs. The default location for this directory is */etc/flume/license/* but it can be changed in the */etc/flume/config* master configuration file.

flexlmLicenseDirectory /etc/flume/license

The license file can have any name but must end in a .lic extension. If more than one .lic file is in the license directory Flume will run if at least one of them is valid for the current date.

NOTE: FlexLM will warn at flume startup that a license is expiring starting two weeks before the expiration date. If a new license is added for a later date but the expiring license is still present, warning messages will still appear until after the expiring license is no longer usable. This is not an error but it will be less confusing if the expiring license is deleted or moved out of the license directory.

Served license files

The Flume license file may be "served." A served license file contains a SERVER line within the file and must be served by your FLEXlm server. Examine your license file to determine if this is required.

If you have a served license file, contact your FLEXlm Server Administrator to have your Flume license file installed. You must also give your FLEXlm Server Administrator the Saratoga Data Systems FLEXlm vendor license daemon file. The vendor daemon file is named *saratoga* and is located in
`<flumeInstallationDirectory>/etc/license/ARCH.linux_26_i86/release/saratoga`

Upon installation, your FLEXlm Server Administrator may have additional set-up instructions for you.

Un-served license files do not contain a SERVER line and do not require the use of a FLEXlm license server.

3.5 The Configuration File

Flume is controlled by setting multiple user configurable parameters stored in a file. The master configuration file must be located either in ***/etc/flume/config*** or in a file pointed to by a symbolic link at that location. These configuration parameters are read by default each time a ***Flume*** client or server is started. The parameters can be overridden from the command line. An alternative configuration file can be specified for a given invocation of the client using the command line switch **–**f as specified below. An example configuration file may be found in the *doc* directory extracted from the Flume installation file. It is also copied to */etc/flume/config* as part of the installation process.

The */etc/flume/config* file is plain text. The format of the */etc/flume/config* file is simply:

```
<parameter-name> <parameter-value>
```

Use only one <parameter-name> <parameter-value> pair per line.

The additional files and directories used by Flume have their locations specified in the master configuration file. These locations are specified by the following keywords and have the default values shown:

The following are text files:

portsFileName	/etc/flume/ports
allowFileName	/etc/flume/allow

The following are directories (note: do not include a trailing slash)

runDirectory	/var/run/flume
logDirectory	/var/log/flume

Automation creation of /etc/flume/config using flumec

The easiest and most reliable way to create or modify the ***Flume*** configuration file is to use the interactive command line program **flumec.** This is a simple text menu program that allows you to specify every configuration parameter supported by the current version of ***Flume***.

```
system@si-centos4x32:~$ flumec
flumec (Flume configuration, v2.3.0) Copyright(C) 2006-2012 Saratoga Data Systems, Inc.
------------------------------------------------------------------------------
 flumec options:
  Config File Param     Cmd Line Option        Value                 Source
  =================     ===============    ===============       ================
  n/a <config file>     (-f)               /etc/flume/config     [default value]
  debug                 (-d)               0                     [default value]
  verbose               (-v)               false                 [default value]
  recurse               (-r)               false                 [default value]
  localInterfaceIp      (-a)               0.0.0.0               [default value]
  summonMethod          (-summon)          ssh                   [default value]
  sshFileName                              ssh                   [default value]
  summonPortDefault                        22                    [default value]
  flexlmLicenseDirectory                   /etc/flume/license    [default value]
  portsFileName                            /etc/flume/ports      [default value]
  allowFileName                            /etc/flume/allow      [default value]
  runDirectory                             /var/run/flume        [default value]
  logDirectory          (-l)               /var/log/flume        [default value]
  tempDirectory                            /tmp                  [default value]
  wholeFile             (-W, -i)           false                 [default value]
```

```
  numericIds           (--numeric-ids)   false                [default value]
  measureIPD                             false                [default value]
  speedMaximum         (-downloadSpeed)  <unspecified>        [default value]
  uploadSpeedMaximum   (-uploadSpeed)    <unspecified>        [default value]
  targetErrorBasisPoints
                       (--targetErrRate) 200                  [default value]
  timeoutThreshold     (-timeout)        3                    [default value]
  summonTimeoutThreshold
                       (-summonTimeout)  3                    [default value]
  synInterval                            200                  [default value]
  compressionLevel     (-cmpLvl)         1                    [default value]
  compressionStrategy  (-cmpStrat)       default              [default value]
  cipherSpec           (-cipherSpec)     none                 [default value]
  MTU                                    1,500                [default value]
  DF                                     true                 [default value]
  includeFiles         (--includeFiles)  {}                   [default value]
  includeDirs          (--includeDirs)   {}                   [default value]
  excludeFiles         (--excludeFiles)  { .* }                [default value]
  excludeDirs          (--excludeDirs)   { .* }                [default value]
--------------------------------------------------------------------------------

Failed to stat prexisting Flume configuration file '/etc/flume/config' because:
        No such file or directory
Attempting to create Flume configuration file '/etc/flume/config' with default settings.
 0 - Set the summon method (summonMethod) the client is to use: ssh
 1 - Set the default summon port number (summonPortDefault) the client is to use: 22
 2 - Set the file name of the ssh client (sshFileName) the client is to use: ssh
 3 - Set the name of allowed ports file (portsFileName) for the client and server: /etc/flume/ports
 4 - Set the allowed ports listed in the portsFileName:         2355-2455
 5 - Set the name of the allowed users file (allowFileName) for the client and server: /etc/flume/allow
 6 - Set the name of the directory where Flume writes runtime data (runDirectory): /var/run/flume
 7 - Set the name of the directory where the Flume server writes its log files (logDirectory): /var/log/flume
 8 - Set the name of the directory where the Flume server writes its temporary files (tempDirectory): /tmp
 9 - Set the name of the directory where you store your local Flume license file (flexlmLicenseDirectory):
/etc/flume/license
10 - Set whether whole files are transferred by default (wholeFile): false
11 - Set whether inter-packet delay is measured by default (measureIPD): false
12 - Set local interface IP address (localInterfaceIp): 0.0.0.0
13 - Set the maximum speed of transmission (speedMaximum) in kilobytes/second: <unspecified>
14 - Set the minimum speed of transmission (speedMinimum) in kilobytes/second: 60
15 - Set the maximum speed of upload (uploadSpeedMaximum) in kilobytes/second: <unspecified>
16 - Set the target packet error rate (targetErrorBasisPoints) in hundredths of a percent: 200
17 - Set the channel timeout threshold (timeoutThreshold) before aborting transfer in minutes: 3
18 - Set the ssh summon timeout threshold (summonTimeoutThreshold) before aborting summoning in minutes: 3
19 - Set the SYN packet interval (synInterval) in milliseconds: 200
20 - Set whether verbose mode is enabled by default (verbose): false
21 - Set debug messaging level (0-9) by default (debug): 0
22 - Set whether recurse mode is enabled by default (recurse): false
23 - Set whether numeric user and group ids are transferred (numericIds): false
24 - Set the compression effort (0-9) as traded off with time (compressionLevel): 1
25 - Set the compression strategy (compressionStrategy) the transfer is to use: default
26 - Set the cipher spec (cipherSpec) encryption method to use: none
27 - Set the maximum transmission unit for datagram packets (MTU): 1,500
28 - Set the don't fragment flag for datagram packets (DF): true
29 - Set the include files glob pattern list (includeFiles):
30 - Set the include directories glob pattern list (includeDirs):
31 - Set the exclude files glob pattern list (excludeFiles):    .*
32 - Set the exclude directories glob pattern list (excludeDirs):       .*
33 - E(x)it configuration without saving:
34 - (S)ave and exit configuration:
Select: (0-34): s
system@si-centos4x32:~$
```

Entering multiple values for include and exclude with flumec

The ***flumec*** program allows entry of multiple values for any of the exclude or include patterns (items 21, 22, 23, and 24 in the menu). After entering a pattern, another pattern may be entered on the next line. Entering return finishes the list of patterns. If only white space is entered on a line a blank line is included in the configuration file unless it is the first or last line of a sequence of patterns.

Manual creation of /etc/flume/config

Optionally, the */etc/flume* directory may be manually created. The permissions need to be set as follows:

```
mkdir /etc/flume
chmod 711 /etc/flume
```

Copy the example ***Flume*** configuration file from the *doc* installation directory to the */etc/flume* directory and set its permissions:

```
cp src/doc/config /etc/flume
chmod 644 /etc/flume/config
```

4. Configuration File Parameters

Many of the configuration file parameters can be overridden by command line switches when invoking the flume client. These are documented in the section entitled ***6.8 Sending and Retrieving*** Options.

4.1 cipherSpec

The *cipherSpec* parameter specifies the algorithm to be used for the inline encryption of both control and data. While the Botan package provides a wide number of choices, we currently only use those that are size preserving. This includes **AES-128/CBC**, **AES-256/CTR**, and **Serpent/CBC**. Encryption can be turned off by specifying **none**.

Default **cipherSpec** is **none**.

4.2 compressionLevel

The *compressionLevel* parameter sets the level of effort for the on-the-fly compression. 0 is off, **1 is default**, up to 9 if enough CPU power is available.

Default **compressionLevel** is **1**.

4.3 compressionStrategy

The *compressionStrategy* allows tuning the compression for specific file types. Valid values are **default**, **filtered**, **huffman_only**, **rle**, and **fixed**.

Default **compressionStrategy** is **default**.

From the zlib package these are (reformatted) notes on using different strategies:

> The strategy parameter is used to tune the compression algorithm. The strategy parameter only affects the compression ratio but not the correctness of the compressed output even if it is not set appropriately.
>
> Use the value Z_**DEFAULT**_STRATEGY for normal data, Z_**FILTERED** for data produced by a filter (or predictor), Z_**HUFFMAN_ONLY** to force Huffman encoding only (no string match), or Z_**RLE** to limit match distances to one (run-length encoding).
>
> Filtered data consists mostly of small values with a somewhat random distribution. In this case, the compression algorithm is tuned to compress them better. The effect of Z_**FILTERED** is to force more Huffman coding and less string matching; it is somewhat intermediate between Z_DEFAULT_STRATEGY and Z_HUFFMAN_ONLY.
>
> Z_**RLE** is designed to be almost as fast as Z_HUFFMAN_ONLY, but give better compression for PNG image data.
>
> Z_**FIXED** prevents the use of dynamic Huffman codes, allowing for a simpler decoder for special applications.

4.4 flexlmLicenseDirectory

The *flexlmLicenseDirectory* configuration parameter tells ***Flume*** the license file location.

Default **flexlmLicenseDirectory** is */etc/flume/license*.

Create the */etc/flume/license* directory and set its permissions:

```
mkdir /etc/flume/license
chmod 755 /etc/flume/license
```

Then copy the ***Flume*** license file to the license directory and set its permissions:

```
cp my.lic /etc/flume/license/saratoga.lic
chmod 644 /etc/flume/license/saratoga.lic
```

Verify that the following line is in your */etc/flume/config* file:

```
flexlmLicenceDirectory /etc/flume/license
```

4.5 DF

This determines the value for the "don't fragment" flag for datagram packets. Value is *true* or *false*.

Default **DF** is **true**.

4.6 excludeFiles

Pattern(s) specifying the names of files to be excluded from transfer unless matching an **includeFiles** pattern. The default pattern is ".*" meaning all "hidden" files (starting with .) are excluded.

Default **excludeFiles** is `".*"`

4.7 excludeDirs

Pattern(s) specifying the names of directories to be excluded from transfer unless matching an **includeDirs** pattern. The default pattern is ".*" meaning all "hidden" directories (starting with .) are excluded.

Default **excludeDirs** is `".*"`

```
excludeDirs ".svn"  (if you need to exclude svn dirs.)
```

4.8 includeFiles

Pattern(s) specifying the names of files to be included in spite of matching an **excludeFiles** pattern.

Default **includeFiles** is `".*"`

4.9 includeDirs

Pattern(s) specifying the names of directories to be included in spite of matching an **excludeDirs** pattern.

Default **includeDirs** is **" . * "**

Exclude and include pattern syntax

Patterns are specified using standard shell wild-card rules where "*" stands for any string and "?" is any character. More precisely: "as described in the Shell and Utilities volume of IEEE Std 1003.1-2001, Section 2.13.1, Patterns Matching a Single Character, and Section 2.13.2, Patterns Matching Multiple Characters."

Exclude and include interaction

When deciding to include a file or directory in a transfer, ***Flume*** always includes it if the name matches any of the include patterns. Then files (or directories) that do not match the include patterns are excluded if they match any exclude pattern.

Multiple patterns

Multiple patterns may be specified for each parameter. The semantics of multiple patterns is "OR" - a match occurs if any of the patterns match.

If they are on the same line, patterns are separated by space or tab. If a space is needed in a pattern the pattern is enclosed in double-quotes ("). If a double-quote is needed within a double-quoted string, two double-quotes in a row are used. Thus the pattern "xyz""a b c" matches the string xyz"a b c.

The exclude and include parameters may also use multiple lines in the configuration file in order to specify multiple patterns. Until another parameter keyword is encountered, subsequent lines that are not comments are treated as additional patterns.

The ***flumec*** program allows entry of multiple values for any of the exclude or include patterns. After entering a pattern, another pattern may be entered on the next line. Entering return will finish the list of patterns. If only white space is entered on a line a blank line is included in the configuration file unless it is the first or last line of a sequence of patterns.

4.10 localInterfaceIp

The IP address of the local network interface.

Default **localInterfaceIp** is 0.0.0.0

4.11 logDirectory

The ***Flume*** server and client each write a log file for every session. The *logDirectory* configuration parameter defines the log file location.

Default **logDirectory** is */var/log/flume*

```
mkdir /var/log/flume
chmod 777 /var/log/flume
```

The configuration parameters file */etc/flume/config* must contain the following line:

```
logDirectory /var/log/flume
```

4.12 MTU

Value is *number of bytes* for the datagram packet.

Default **MTU** is *1500*

4.13 portsFileName

The *portsFileName* configuration parameter tells ***Flume*** the name of a file that specifies the local port numbers you have configured for Flume data transmission.

Default **portsFileName** is */etc/flume/ports*

```
portsFileName /etc/flume/ports
```

A decision must be made on the port numbers for ***Flume*** to use on each computer. Any set of port numbers may be selected, but they may not be in conflict with other network applications and services. A different set of port numbers may be used on each computer.

For example, use ports 2355 through 2455 for ***Flume*** communication on all computers.

First, open ports 2355 through 2455 for bi-directional TCP and UDP communications in all firewalls protecting all the computers you intend to use for ***Flume***.

Second, if network address translation (NAT) is in use between a computer running ***Flume*** and the network used to reach the other (for instance, the Internet), the router or gateway must forward all ports that may be used by ***Flume*** from the "outside" address to the local address (typically private such as 192.168.1.88) of the computer running ***Flume***. While strictly speaking only one TCP and one UDP port need be forwarded for a given transfer session, in practice forwarding both TCP and UDP for the entire range included in /etc/flume/ports is recommended.

Tell ***Flume*** the local ports to use by creating a plain text file named */etc/flume/ports* with the following contents:

```
2355-2455
```

Set permissions on the */etc/flume/ports* file:

```
chmod 644 /etc/flume/ports
```

The format of the */etc/flume/ports* file is a contiguous range of integer numbers separated by "-"(a dash with no intervening white spaces) or a single port number on each line. For example:

```
2355-2395
3000
2500-2600
```

tells ***Flume*** to use ports 2355 through 2395 (inclusive), port 3000, and ports 2500 through 2600 (inclusive).

4.14 restartCountDefault

The default number of times a session restart should be allowed to be attempted per check-point file.

Default **restartCountDefault** is 1,000,000

4.15 runDirectory

Flume keeps track of data using several files. The location of these runtime files is defined by setting the *runDirectory* configuration parameter.

Default **runDirectory** is */var/run/flume*.

As super-user (i.e., root) create the */var/run/flume* directory and set its permissions:

```
mkdir /var/run/flume
chmod 771 /var/run/flume
```

Create the *ports* file in run directory:

```
touch /var/run/flume/ports
chmod 666 /var/run/flume/ports
```

Make the *check_points* sub-directory in the run directory and set permissions:

```
mkdir /var/run/flume/check_points
chmod 777 /var/run/flume/check_points
```

Confirm that the configuration parameter file */etc/flume/config* contains the following line:

```
runDirectory /var/run/flume
```

4.16 speedMaximum

The maximum speed of transmission of ***Flume*** is configured in the */etc/flume/config* file. The value is an integer number in kilobytes per second (KB/s). It is important that ***Flume's*** maximum transmission speed is limited to the bandwidth of the client-server network connection. The maximum speed may be set to any value less than the bandwidth of the client-server network connection.

Default **speedMaximum** is 0

Example: if the client-server network connection is 8 Megabits per second (Mbps), the bandwidth in kilobytes per second (KB/s) is 1000. If you intend to have ***Flume*** attempt to use all available bandwidth, put the following line in the */etc/flume/config* file:

```
speedMaximum 1000
```

If ***Flume*** is limited to half the bandwidth of the connection:

```
speedMaximum 500
```

At the beginning of a session, the ***Flume*** client and server negotiate the maximum speed for the session. This negotiated session maximum speed is the lesser of the value of the *speedMaximum* parameter on the server-side and the value of the *speedMaximum* parameter on the client-side. For example, if the *speedMaximum* parameter is set to 1000 in the ***Flume*** client */etc/flume/config* file and is set to 500 in the server */etc/flume/config* file, a ***Flume*** session between these two computers will be limited to the lesser of the two values, in this case: 500 KB/s.

You may also set the value of the *speedMaximum* parameter to 0, which tells ***Flume*** that you have not specified a value and to use the *speedMaximum* value specified at the other end of the ***Flume*** session. For example, if the value of the *speedMaximum* parameter is 1000 in the server's */etc/flume/config* file and the value of the *speedMaximum* parameter is 0 (unspecified) in the client's */etc/flume/config* file, the negotiated session's maximum speed will be 1000 KB/s.

If both ends have *speedMaximum* set to 0, ***Flume*** defaults to 250 KB/s.

If the ***Flume*** server and the ***Flume*** client have an unspecified speedMaximum parameter (i.e., speedMaximum 0 on both ends of a Flume connection) the default maximum speed will be 250KB/s.

It is important to note that ***Flume*** coexists with your other TCP network traffic. Flume uses all available bandwidth of the connection: the bandwidth that is not already allocated to TCP network traffic.

NOTE: Starting with release 2.3.0.2408 speedMaximum, uploadSpeedMaximum, and speedMinimum can values can have a decimal point in them in order to specify non-integer values, including values less than 1 KB/s. For example speedMaximum 0.1 is .1 KB/s or 800 bits/sec.

4.17 uploadSpeedMaximum

The *speedMaximum* parameter sets the speed regardless of the direction of transfer (to or from the server). Some networks (such as ADSL or cable modem connections) have different bandwidth depending on the direction of transfer –

generally with higher speed allowed incoming than outgoing. Only when this is the case, a separate *uploadSpeedMaximum* should be specified to define that differing outgoing speed.

Default **uploadspeedMaximum** is 0

For instance for a cable modem with 16 Mbps (2 MB/s) incoming and 2 Mbps (250 KB/s) outgoing the following parameters would be appropriate:

```
speedMaximum 2000
uploadSpeedMaximum 250
```

4.18 sshFileName

The path name of the *ssh* client application executable program file. The default value is *ssh* and assumes that *ssh* can be found in your *PATH* environment variable.

Default **sshFileName** is ssh

If *ssh* cannot be found in your *PATH*, set this variable to the executable program file of *ssh*.

sshFileName file

4.19 summonMethod

Use the ***Flume*** client to initiate all file transfers. The ***Flume*** client must summon the ***Flume*** server on the remote computer. ***Flume*** supports three methods of summoning the ***Flume*** server: *ssh*, *inetd/xinetd*, and *page*. The *summonMethod* configuration parameter tells the ***Flume*** client which method to use to remotely summon the ***Flume*** server.

Default **summonMethod** is ssh

Using ssh summoning

The *ssh* method is the most secure method of summoning the ***Flume*** server. It uses secure shell to log into the remote computer and start the ***Flume*** server. To use the *ssh* summoning method, simply put the following line in the */etc/flume/config* file:

```
summonMethod ssh
```

Using xinetd summoning

The *inetd/xinetd* method is the standard facility for remotely starting services. The *inetd/xinetd* summoning method requires that a port dedicated to summoning ***Flume*** via *inetd/xinetd* is allocated and opened in your firewall.

Here is an example:

Dedicate port 2354 (other ports can be used) for *inetd/xinetd* summoning of the ***Flume*** server on all computers on which ***Flume*** will be used. Open port 2354 in your firewall for bi-directional TCP communication and put the following lines in the */etc/flume/config* file:

```
summonMethod inetd
summonPortDefault 2354
```

Configure *inetd* or *xinetd* to start the ***Flume*** server when contact is made on port 2354. As an example, assume the use of *xinetd* as opposed to *inetd*. To configure *xinetd* to start a ***Flume*** server, add a file named */etc/xinetd.d/flume* to all computers on which ***Flume*** will be used. The content of file */etc/xinetd.d/flume* is plain text as follows:

```
service flume
{
        socket_type = stream
        protocol = tcp
        wait = no
        user = root
        server = /usr/local/bin/flumed
        port = 2354
        disable = no
}
```

Add the ***flume*** service to your system by adding the following line to the */etc/services* file:

```
flume 2354/tcp
```

Restart the *xinetd* service by issuing the command:

```
/etc/init.d/xinetd restart
```

Using page mode summoning

Page mode summoning is a pre-summoned mode. In high latency connections when there will be multiple transmissions of data this allows the delay in starting a connection to be avoided. This is started with a new command

`flumep` `[<username>]@`**`<remotehost>:`**`[<port>]`

to establish a proxy server with a running connection to the ***flumed*** server on a remote system. Once ***flumep*** is running, transfers can be done from the ***Flume*** client by specifying

`-summon=page:<port>.`

The <username> parameter is optional on ***flumep*** and defaults to the current user. <port> is also optional and defaults to 2317.

The ***flumep*** proxy server can be on the same system from which ***Flume*** will be run or on a separate system. If a separate system is used for ***flumep***, it should be “close” to the flume client system - ideally in the same LAN.

If the three systems are **localhost**, **proxyhost**, and **remotehost** the syntax is:
proxyhost> flumep @remotehost:2317

localhost> flume -summon=page:2317 FileToSend proxyhost:/location

And with just two systems **localhost** and **remotehost** the full syntax is:
localhost> flumep @remotehost:
localhost> flume -summon=page:2317 FileToSend localhost:/location

Or **localhost** can be left out:
localhost> flumep @remotehost:
localhost> flume -summon=page:2317 FileToSend :/location

4.20 summonPortDefault

This parameter is the port to use for an outbound connection from a client to a server. This parameter forces the server to start the appropriate daemon.

Default **summonPortDefault** is 22

Therefore for ssh summoning it would typically be set to port 22 (though many now use an alternative port for ssh such as 25, 27, or 29). This parameter is used for all forms of summoning.

NOTE: the flumec configuration program offers to set summonPortDefault (to an appropriate standard value) if you change the summonMethod.

4.21 summonTimeoutThreshold

The *summonTimeoutThreshold* parameter specifies how many minutes to wait when summoning the ***flumed*** server before aborting.

Default **summonTimeoutThreshold** is 3

4.22 targetErrorBasisPoints

This parameter allows tuning of the PID controller that is used to govern the sending rate. This is specified in 1/100 of 1% increments. The default value is 200 which equates to 2%.

Default **targetErrorBasisPoints** is 200

4.23 timeoutThreshold

The *timeoutThreshold* parameter specifies how many minutes to wait during data transmission before aborting due to a lost (or never established) connection. It also specifies the time to wait during the initial summoning unless overridden by the *summonTimeoutThreshold*.

Default **timeoutThreshold** is 3

4.24 tempDirectory

This parameter gives the directory where ***Flume*** writes temporary files. The default value is that specified by your *TMPNAM* environment variable, or its default */tmp.*

Default **tempDirectory** is /tmp

4.25 verbose

The amount of data in log files. A value is *true* or *false*. The default is *false*. A value of *true* causes the ***Flume*** client and server to be more verbose in their logs.

Default **verbose** is *false*

When ***Flume*** is run in verbose mode the following listing of configuration parameters and their origin will go to both the console and the client.log file. This same list is shown when running ***flumec*** if verbose mode is enabled.

```
   Config File Param    Cmd Line Option        Value                Source
 ================    ===============   ==============       ===============
 n/a <config file>    (-f)              /etc/flume/config      [default value]
 debug                (-d)              0                      [default value]
 verbose              (-v)              false                  [config file]
 recurse              (-r)              false                  [default value]
 localInterfaceIp     (-a)              0.0.0.0                [default value]
 summonMethod         (-summon)         ssh                    [default value]
 sshFileName                            ssh                    [default value]
 summonPortDefault                      22                     [default value]
 flexlmLicenseDirectory                 /etc/flume/license     [default value]
 portsFileName                          /etc/flume/ports       [default value]
 allowFileName                          /etc/flume/allow       [default value]
 runDirectory                           /var/run/flume         [default value]
 logDirectory         (-l)              /var/log/flume         [default value]
 tempDirectory                          /tmp                   [default value]
 wholeFile            (-W, -i)          false                  [default value]
 numericIds           (--numeric-ids)   false                  [default value]
 measureIPD                             false                  [default value]
 speedMaximum         (-downloadSpeed)  10,000                 [config file]
 uploadSpeedMaximum   (-uploadSpeed)    <unspecified>          [config file]
 targetErrorBasisPoints
                      (--targetErrRate) 200                    [default value]
 timeoutThreshold     (-timeout)        3                      [default value]
 summonTimeoutThreshold
                      (-summonTimeout)  3                      [default value]
 synInterval                            200                    [default value]
 compressionLevel     (-cmpLvl)         1                      [default value]
 compressionStrategy  (-cmpStrat)       default                [default value]
 cipherSpec           (-cipherSpec)     none                   [default value]
 MTU                                    1,500                  [default value]
 DF                                     true                   [default value]
 includeFiles         (--includeFiles)  {}                     [default value]
 includeDirs          (--includeDirs)   {}                     [default value]
 excludeFiles         (--excludeFiles)  {}                     [config file]
 excludeDirs          (--excludeDirs)   { .* }                 [default value]
-------------------------------------------------------------------------------
```

5. Networking

5.1 Wide Area Networks

Wide Area Networks - Types and suitability

Flume can make a significant difference in the performance of transfers on long distance networks where the latency is due to the physical separation.

Some common network types:

1. **The public Internet** – typically with systems behind routers, but systems can be used on public IP addresses if properly configured and hardened. See our short paper on Security Considerations with Public Systems. ***Flume* can do very well over the Internet**, especially with a high bandwidth connection out of the sending side and into the receiving side.

2. **Multi Protocol Label Switching** (MPLS) is a technique that uses the same backbone networks as the Internet, but provides private end to end connections with guaranteed bandwidth. This is the most common type of corporate WAN and **works very well with *Flume***.

3. **Dedicated Leased Line** is a completely private circuit between company sites. We have tested ***Flume*** on a 45 Mbps leased line with 275 ms of latency between California and India, with **extremely good results**.

4. **Frame Relay (FR)** is an older form of private wide area networking with less predictable performance. A given FR circuit shares bandwidth from a pool which totals less than 100% of the combined circuit bandwidths of all the customers. ***Flume* may or may not benefit** an FR connection, depending on the activities of other customers.

5. **Virtual Private Network** (over the public Internet) – also known as tunneling - typically encapsulates all traffic in a TCP stream. This encapsulation requires end to end handshakes regardless of traffic. ***Flume* may not even run over VPN,** and even if it does run, it will not be able to accelerate traffic.

Wide Area Networks – possible issues

In addition to the underlying type of WAN, other issues may impact the ability of ***Flume*** to accelerate the connection.

1. **Bottlenecks** – Some networks have fast segments joined by slower ones. For example a T3 (45 Mbps) may connect to a T1 (1.5 Mbps) and back to

a T3. Even though it looks like a T3 at each end, the throughput can never exceed that of the T1 link. On a high bandwidth, high latency connection, TCP throughput is generally much lower than the bandwidth of the connection. For example, assume a dedicated 2M bits/second (2 Mbps) network where TCP-based (e.g. FTP) files transfer at 40K Bytes/seconds, (40 KB/s) or only 320Kbps. On such a dedicated network connection, ***Flume*** will demonstrate throughput approaching 250KB/s or the entire capacity of the 2Mbps connection.

2. **Quality of Service (QoS) control** – This refers to prioritization that may be done in the routers or gateways. For instance some systems may lower the priority of UDP (or **throttle** it) to increase throughput of TCP traffic.

3. **Accelerator Appliances** – These are special-purpose dedicated computers designed to accelerate *applications* over a WAN. Typically these use caches at the remote end and a strategy for keeping information in the cache that is expected to be accessed often. These devices also focus on specific application acceleration, most often email and intranet internal web access. ***Flume*** **will typically work much better if any such appliance is bypassed for** ***Flume*** **transfers.**

1.2 Basic Flume Network Requirements

This can be summarized in one sentence: **The two systems must be able to "see" each other over the network being tested.**

This means several things:

1. They can **ping** each other.

   This should be verified and the ping times noted as part of the evaluation process.

2. If **ssh** is used as the invocation method (recommended), it must be possible to make an ssh connection from the local / client system to the remote/server system.

   This should be tested with a standard ssh connection – using the same username as will be used for ***Flume*** operation. In addition to verifying the ability to connect to the remote system, once connected you should verify the ***flumed*** program is in the search PATH by typing:

   ```
   which flumed
   ```

3. All **ports** that will be used by ***Flume*** are both **open** (not blocked by a firewall) and **routed** (if the two systems are not in the same IP address space).

5.3 Firewall and router requirements

The ***Flume*** client and server use two ports to communicate: typically starting with 2354 and 2355, although a pool of ports from which to pick two is specified by default in the file */etc/flume/ports*. Ports currently in use by ***Flume*** are tracked by default in the file */var/run/flume/ports*. [This allows multiple copies of ***Flume*** to run simultaneously.]

Flume uses the **first port** as its **control** channel. This port must be open for TCP communication through your firewall(s). ***Flume*** must be able to accept inbound TCP connections as well as initiate outbound TCP connections on this port. For instance, in the default configuration with no other ports already in use, this will be port 2354.

Flume uses the **second port** as its **data** communication channel. Open the second port for UDP communication through your firewall(s). ***Flume*** must be able to accept inbound UDP connections as well as initiate outbound UDP connections on this port. For instance, in the default configuration with no other ports in use, this will be port 2355.

You must open all ports that can be used by ***Flume*** through your firewall(s) in order for ***Flume*** to operate.

If network address translation (NAT) is in use such that either end of the connection is on a private IP address, the router(s) providing NAT must be set to forward any ports being used by ***Flume*** to the private IP address for that system.

While not strictly necessary, it is easiest to open UDP and TCP on all required ports.

5.4 Configuring Flume network speed

You must determine how much of your network bandwidth to dedicate to ***Flume*** for file transfers.

Do not specify a maximum speed that is extremely greater than your network can support. Because of the aggressive way ***Flume*** sends data, this can overwhelm the connection and control channel and cause the session to fail. In extreme cases, it can even knock the connection “off the air.”

It is best to start with a number that you know is small enough, verify the ability to transmit data, and then increase *speedMaximum* until the best performance is achieved.

For example, assume a 2 Mbps connection between two computers. Further assume that the entire bandwidth will be dedicated to ***Flume*** file transfers. ***Flume*** must be configured to transfer files at 250 KB/s. Please note the difference in units: 2 Mbps (2M **bits**/second) and 250 KB/s (250K **Bytes**/second) are the same.

If ***Flume*** will operate over a shared network connection, account for other network traffic in setting ***Flume's*** maximum transmission speed.

Experimentation may be required to determine the best maximum transmission speed setting for ***Flume*** over your network connection. If the ***Flume*** maximum transmission speed is set too low, ***Flume*** will not operate at peak efficiency. If the ***Flume*** maximum transmission speed is set too high, Flume will adjust its transmission speed down to the available capacity, but it will also periodically attempt to increase its transmission speed, possibly causing your network to discard packets and Flume to retransmit more packets than it would otherwise.

The ***Flume*** maximum transmission speed is set using the ***speedMaximum*** configuration parameter. The units of this parameter are **kilobytes per second (KB/s)** not kilobits per second (Kbps).

Do not specify the units in a Flume parameter configuration file. Only supply the value of the parameter. For example, if you wish to set Flume's maximum transmission speed to 250 KB/s, place the following line in the configuration file:

```
speedMaximum 250
```

You may specify a value of the *speedMaximum* parameter in */etc/flume/config* at both client and server ends of the connection.

If you specify a different value for each end, the lesser of the two values will be used. For example, if you specify a value of 200 for the *speedMaximum* parameter in */etc/flume config* file on the server (***flumed***) end and specify a value of 300 on the client (***flume***) end, then the maximum transmission speed will be 200 KB/s.

The default value for *speedMaximum* is 0 (zero), which indicates that the value is unspecified by you. If one end of the transmission, either ***flumed*** or ***flume***, has an unspecified value for the *speedMaximum* parameter, then the value specified at the other end of the transmission will be used. If the *speedMaximum*

parameter is unspecified at both ends of the transmission, then a value of 250 KB/s (2 Mbps) is used.

A parameter called ***uploadSpeedMaximum*** allows specification of a different value for cases such as ADSL or cable modems where the network is asymmetric and provides different bandwidth in each direction. This can also be controlled via a command line switch for either the upload or download speed.

Recommended best practice for most evaluations: set *speedMaximum* and *uploadSpeedMaximum* to 0 on every system but the one from which tests will be run. Then set *speedMaximum* on that one system only. If and only if the connection is a different speed outgoing than incoming, also set *uploadSpeedMaximum* on that same system, otherwise also set it to 0.

6. Evaluation

Flume should be evaluated on **a high bandwidth, high latency network connection**. This will demonstrate the superior throughput of ***Flume*** relative to TCP. ***Flume*** can be set up and run on a local area network (LAN) or even on a single system to ensure that it is set up and configured properly. But this will not demonstrate any of the benefits of ***Flume***.

It is possible to do this with a network simulator but we recommend the use of an actual connection in which you are currently unable to make use of the available bandwidth. Note also that a simple simulation of limited bandwidth may not predict the real-world ***Flume's*** superiority.

On a high bandwidth, high latency connection, you should experience TCP throughput that is much lower than the bandwidth of the connection. For example, assume a dedicated 2M bits/second (2 Mbps) network where TCP-based (e.g. FTP) files transfer at 40K Bytes/seconds, (40 KB/s) or only 320Kbps. On such a dedicated network connection, ***Flume*** will demonstrate throughput approaching 250KB/s or the entire capacity of the 2Mbps connection.

6.1 Checklist for Testing Flume

Flume evaluation needs to reflect your particular needs and environment. The following checklist can ensure a smoother evaluation:

- ❑ Qualification – is ***Flume*** appropriate for your network?
  - o What kind of connection do you have?
  - o What is the supposed bandwidth?
  - o What latency (**ping** time) does it have?
  - o What transfer rate do you get today with:
    sftp
    ftp
    scp
    rsync --whole-file

    NOTE: running at least one of these will both (1) provide an initial value to use for the ***Flume*** *speedMaximum* parameter in */etc/flume/config* and (2) provide a baseline for comparison.

- ❑ Additional computer diagnostics – at each end
  - o **uname -a** to get kernel information
  - o **cat /etc/redhat-release** for Red Hat or CentOS systems
  - o **cat /proc/cpuinfo** to get processor speeds and number of CPUs / cores

- **df –h** to get disk space available
- **/sbin/traceroute** between local and remote system
- **fdisk -l** to get disk partition information (requires root access)

- ❑ Ensure network connectivity:
  - exists between the systems to be evaluated
  - all ports made available to ***Flume*** (in */etc/flume/ports*)
  - verified as open (not blocked by firewall)
  - routed (if necessary). This can be tested with netcat (nc).

- ❑ Complete the additional recommended system administration setup discussed in Appendix A to simplify testing.

- ❑ Get all necessary files from Saratoga Data Systems, Inc.

- ❑ Installation and configuration – on each end

- ❑ Basic verification of installation from local / client end:
  - *install and configure localSystem*
  - *verify the **Flume** executable is findable*
  - localSystem$ **which flume**
  - *verify that you can ssh to the remote system*
  - localSystem $ **ssh remoteSys**
  - *install and configure flume at remoteSys and verify flumed can be invoked after an ssh connection*
  - localSystem $ **ssh remoteSys**
  - remoteSys$ **which flumed**

- ❑ First very basic test run – local system only
  - *copy a file locally entirely using flume*
  - localSystem$ **flume /etc/flume/config localhost:/tmp/**
  - *this will test the ability to access the license file and copy the configuration file to the local /tmp directory*

- ❑ Second very basic test – copy to remote system
  - localSystem$ **flume /etc/flume/config remoteSys:/tmp/**

- ❑ Third very basic test – copy from remote system
  - localSystem$ **flume remoteSys:/etc/flume/config /tmp/**

- ❑ Fourth test – quick test of incremental operation
  (with 0 length file at remote)
  - localSystem$ **ssh remoteSys touch /tmp/config**
  - localSystem$ **flume /etc/flume/config remoteSys:/tmp/**

- ❑ In case of difficulty, contact Saratoga Data Systems, Inc. and provide all possible information.
  - o terminal output from installation
  - o results of first, second, and third tests
    terminal output from client and log files from server

- ❑ Begin evaluation testing

6.2 An example evaluation plan

Your evaluation plan needs to meet your needs. However, we recommend you follow a plan resembling this:

1. Characterize network: type, ping, bandwidth available, and transfer rate achieved with TCP.

2. Install, configure, and do basic verification of ***Flume*** at each location as shown above.

3. NOTE: If any ***Flume*** transfers fail, please enable additional debugging information with **-d=8** in the command line, rerun the transfer, and send us the console output, client log, and server log.

4. Transfer files of increasing size up to your needed sizes.
   a. Start with smaller files : 1 to 10 MB
   b. Calibrate network with FTP or other TCP protocol
   c. CRITICAL: Set initial */etc/flume/config* ***speedMaximum*** based on initial TCP speed. You will need to try higher values later.
   d. Once successful with ***Flume*** transfer, increase speedMaximum and record results until no increase in transfer speed is seen. This may take multiple runs due to variations in your network.
   e. If your maximum bandwidth is *NNN* KB/s (8 x NNN Kbps), the *speedMaximum* parameter can often be set as high as 1.5X *NNN* to push ***Flume*** to maximum performance. In other words, if *NNN* is 1000 KB/s, you may need to set *speedMaximum* as high as 1500 for best results.
   f. For testing purposes, we recommend you leave *speedMaximum* 0 at the remote / server end and control it from the local /client end.

   For asymmetric connections, you must set *uploadSpeedMaximum* **differently from *speedMaximum*.**

5. Testing the incremental update capabilities of ***Flume*** requires some forethought.
    a. The simplest test is to use touch to create a zero length file with the name of the file to be transmitted. This will cause ***Flume*** to analyze and transmit the differences between the source files and the empty file.
    b. A better, but still artificial, test is to take a large file, transmit it to the destination but interrupt the transfer at various points and save each partial version. Then you can run transfers of the real file with varying amounts already present.
    c. A real test of incremental update is to have an entire directory tree of files, some of which are newer at the source, and compare ***Flume*** with -W (where all the files are transmitted in their entirety) against a separate run using another copy of the destination files where incremental transfers take place. Unless the files are very large and mostly different, the incremental mode will usually take substantially less time.
    d. If your test network has latency higher than a local area network (LAN) or you choose to inject delay with a network simulator, you should compare rsync --whole-file with flume -W and rsync incremental with ***Flume*** incremental. In the presence of latency higher that 100 ms, ***Flume*** should substantially outperform rsync.
6. For each test we recommend that you note:
    a. File size and type (although ***Flume*** is not affected by file type)
    b. Speed and time with TCP
    c. Speed and time with Flume at each setting of speedMaximum
    d. If the network is highly variable we recommend you record the ping time before and after each test.

6.3 Ensuring the Flume Connection by Verifying Ports

Independently verify that ***Flume*** ports are open on Unix/Linux using the *nc* program. The *nc* program allows testing ports by sending text packets from one computer to another. Note: on some Unix/Linux releases the *nc* program may be named *netcat*. If your system has *netcat* instead of *nc*, please read the online *netcat* manual (command: *man netcat*) for a detailed description of its operation.

Receiving End

On the receiving end, instruct *nc* to listen to a port for a TCP connection from another computer as follows:

```
Unix> nc –l <host> <port>
```

Where <host> is the domain name or IP address of the sending computer and <port> is the port number on which *nc* is to listen.

To instruct *nc* to listen to a port for a UDP connection, modify the above command as follows:

```
Unix> nc –l –u <host> <port>
```

Sending End

Once *nc* is listening on the receiving end, use *nc* to send a TCP packet from the sending computer as follows:

```
Unix> echo “hello there” | nc <host> <port>
```

Where <host> is the domain name or IP address of the remote (receiving end) and <port> is a local Flume port number.

Again, to send a UPD packet to a listening *nc* program on a remote computer, modify the above command as follows:

```
Unix> echo “hello there” | nc –u <host> <post>
```

If the connection is open for a particular protocol (TCP or UDP), *nc* will print “hello there” to the terminal at the receiving end. If the connection is not open, *nc* will not print anything.

Caveat: some versions of *nc* require you to proceed the port number (i.e., the <port> number argument) with ‘-p’.

Use this technique to verify the connection through the ports to use with ***Flume*** in both directions and with both the TCP and UDP protocols.

6.4 Running Flume

All ***Flume*** file transfers (sending or receiving) are initiated from the local / client side which can either push files to or pull files from the remote / server end of the connection. Specify the files to copy on the command line when invoking the Flume client.

NOTE: do not manually start ***flumed*** at the remote / server before running ***flume*** at the local / client.

Flume invocation

From the local / client end, the ***flume*** program establishes a connection to the remote / server end and invokes ***flumed***. This is done using the following mechanisms.

1. ssh – recommended
2. inetd – may be more convenient

Note: ssh invocation is via whatever port is used at the remote / server end for normal ssh. The default is port 22 but, when the remote is located on a public Internet connection, many sites change this to another value such as 24 or 29. This is normally set at the remote end in /etc/ssh/sshd_config and controlled at the local end in /etc/ssh/ssh_config The flume configuration file can be used to change the ssh port that Flume will use or it can be specified at the command line used to start a Flume transfer.

6.5 Flume Client Command Line

Remote File Specifications

The ***Flume*** client requires that remote files and directories be specified with the following syntax:

```
[<user>@]<host>:<remote-path>
```

Where `<user>` is a valid user login name on the remote computer identified by `<host>` and `<host>` is the domain name or the network IP address of the remote computer. The square brackets around `<user>@` indicates that it is optional. If you don't include `<user>@` in a remote path specification, `<user>` defaults to the login of the current user on the local host.

Finally, `<remote-path>` is a path name on the remote computer identified by `<host>`.

Source vs. Destination

Regardless of whether you are copying files to a remote computer or from a remote computer, you must think in terms of the source path and the destination path in order to accurately predict the resultant file name(s) at the destination.

Firstly, the source path must always exist. It may be a file or a directory, but it must exist.

However, the destination path may or may not exist. If the destination path does not exist, the copied files or directories are renamed to the given non-existent name.

If the destination path exists and is a directory, the source files will be copied to the destination directory. If the destination path exists and is a file, the source file is copied with the new destination name.

Source vs. Destination with pipes

Flume can be used in a piped series of commands. The use of "**-**" (dash, hyphen or minus character) as the source or destination path tells ***Flume*** to either get data from stdin or output data to stdout. In order for this to be meaningful at the remote (server) end, it is also necessary to supply the command to be run at the remote (server) to produce or consume the piped data data.

The following combinations are now supported

- local source path **-**
  means that data must be piped into ***Flume***
- remote source path **-**
  means that **--remoteCmd** must provide data on standard out for ***flumed*** to send send back to the client
- local destination path **-**
  means that the stdout from ***Flume*** must be redirected to a file or piped to a command for which it is meaningful
- remote destination path **-**
  means that **--remoteCmd** must expect input on stdin that will come from ***flumed*** stdout.

See the Examples section below showing the use of tar with ***Flume*** piping either to send or receive data.

6.6 Sending Files to a Remote Computer

To send local files to a remote computer on which ***Flume*** is installed and licensed, use the following syntax:

```
flume [option]… <local-path>… [<user>@]<host>:<remote-path>
```

You may specify one or more files and directories with `<local-path>`. ***Flume*** sends the list of files and directories `<local-path>`… to `<host>` as `<user>` with `<remote-path>` as the root of the destination copy.

6.7 Retrieving Files from a Remote Computer

To retrieve files from a remote computer on which ***Flume*** is installed and licensed, use the following syntax:

```
flume [option]… [<user>@]<host>:<remote-path> <local-path>
```

Flume retrieves `<remote-path>` from `<host>` as `<user>` with `<local-path>` as the root of the destination copy.
Please note an important difference in capability between sending and retrieving files: the sending syntax allows the specification of any number of local files and directories as the copy source. The retrieving syntax allows the specification of exactly one source file or directory.

6.8 Sending and Retrieving Options

The sending and retrieving syntax accepts the following options:

-p: make the **permissions** of the copied destination files the same as the source files.

-t: make the **timestamps** of the copied destination file the same as the source files (local time).

-r: **recursive** mode, if a source is a directory, recursively copy the directory and all files and sub-directories beneath the directory.

-u: **update** mode, copy only the source files that correspond to non-existent destination files and destination files with modification timestamps older than the corresponding source file (i.e., it does not overwrite newer files). Note: this update option implies the use of the –t option and that, if the destination files exist, their modification times are meaningful (i.e., they were previously transmitted using ***Flume*** with the –t or –u option).

-rc <number>: **restart count**, set the number of allowed restarts of interrupted session. If –rc is not specified, the value of *restartCountDefault* ***Flume*** configuration parameter is used. The default value for the *restartCountDefault* configuration parameter is 1,000,000.

-v: turn on **verbose** logging.
-d: turn on **debugging** messages.

-cmpLvl: Level of effort for on-the-fly compression. 0 is off, 1 is default, up to 9 if enough CPU power is available.

-cmpStrat: Compression strategy valid values are, **default**, **filtered**, **huffman_only**, **rle**, and **fixed**.

--excludeFiles: Overrides the value(s) of **excludeFiles** in the config file. Pattern(s) specifying the names of files to be excluded from transfer unless matching an **includeFiles** pattern.

--includeFiles: Overrides the value(s) of **includeFiles** in the config file. Pattern(s) specifying the names of files to be included in spite of matching an **excludeFiles** pattern

--excludeDirs: Overrides the value(s) of **excludeDirs** in the config file. Pattern(s) specifying the names of directories to be excluded from transfer unless matching an **includeDirs** pattern.

--includeDirs: Overrides the value(s) of **includeDirs** in the config file. Pattern(s) specifying the names of directoriesto be included in spite of matching an **excludeDirs** pattern.

--remoteCmd: Used with Flume **piping** only when the <remote-file-name> is specified as **-**. This specifies a command or series of commands to be executed at the server. If flume is pushing data to the server this should consume data via stdin that is coming from flumed stdout. If flume is pulling data from the server this should provide data on stdout for piping into flumed stdin.

--targetErrRate: Allows tuning of the PID controller that is used to govern the sending rate. This is specified in 1/100 of 1% increments. The default value is 200 (2%).

-cipherSpec: the algorithm used for the inline encryption. Valid values are **AES-128/CBC**, **AES-256/CTR**, **Serpent/CBC** and **none**.

WARNING: Command line exclude and include are not used when retrieving files

The patterns at the sending end of a ***Flume*** transfer control to which files are excluded or included. This means that when the ***Flume*** client is used to send files, the command line parameters are meaningful. When the ***Flume*** client is used to retrieve files, the exclude and include parameters in the configuration file at the server control the transfer. If command line values for exclude or include are supplied when retrieving files, a WARNING is issued that they are not used. Example:

```
[root@c64-centos5x64 ~]# flume -r -v --excludeDirs "*hid*" public:/tmp/915x/ /tmp
flume (Flume client, v2.2.1) Copyright(C) 2006-2009 Saratoga Data Systems, Inc.
Warning: Exclude directories option (--excludeDirs) has no effect in receive mode.
```

Exclude and include allowed multiple times on a command line

The exclude and include command line parameters can be used more than once on a command line to specify multiple patterns.

6.9 Check-point & Restart

The ***Flume*** client periodically saves the state of the currently running session so that if the session fails to complete (e.g., due to a network failure) you can restart the session from the last check-point.

The ***Flume*** client allows the following operations on check-point files.

List your existing check-point files

```
flume –ckpList
```

The –ckpList command line option instructs the ***Flume*** client to list all check-point files owned by the current user that are not in use by another ***Flume*** client session.

Clear (delete) a check-point file

```
flume –ckpClear [<check-point-file>]…
```

With no <check-point-file> specified, -ckpClear instructs the ***Flume*** client to delete all existing and unused checkpoint files for the current user. With one or more <check-point-file> arguments, ***Flume*** deletes the named checkpoint files provided they are owned by the current user and not in use by another ***Flume*** client session.

Restart from a check-point file

```
flume –ckpRestart <check-point-file>…
```

-ckpRestart instructs the ***Flume*** client to sequentially restart sessions from the given list of check-point files, provided each is owned by the current user and not in use by another ***Flume*** client session.

6.10 Progress reports and logs

The *flume* program reports its progress to the terminal (on *stderr* because using ***Flume*** in a pipe requires output of the data to *stdout*) as files transfer. ***Flume*** also reports the total bytes, overall KB/s, and any savings achieved by the use of incremental transfers. At both the local / client and remote / server end, a log file is written in */var/log/flume/* (by default). Client logs include .client. in the pathname. The log file name is constructed from the date and time of the transfer.

6.11 Using the Saratoga Data Systems, Inc. public servers

If you can get to and from the Internet with one or more of your test systems and have an appropriate set of ports (at least two) routed from the Internet to your test system, we have servers in Santa Clara, California (public.saratogadata.com) and in Shanghai, China (flume-sh.saratogadata.com). Each of these has a 100 TX full-duplex Ethernet connection to the Internet at 100 Mbps = 12,500 KB/s. The speedMaximum parameter on each is currently set to 5000 (KB/s) = 40 Mbps.

If you want to try ***Flume*** using these systems we will supply you with a login. The Santa Clara system also has anonymous ftp (read-only) access to a set of test files that can be transferred for calibration and then for comparison. We currently have these files in /disk2**/ftp/pub/test-data** (via scp) or ftp://public.saratogadata.com/pub/test-data/ (via FTP)

- 2M.taz
- 15M.tar
- 250M.taz
- 500M.iso
- 2G.iso

As of November, 2012 – the default configuration at public.saratogadata.com is Flume 2.3.

You can confirm the version of Flume installed at public.saratogadata.com with the following command:
$ **ssh yourUserName@public.saratogadata.com flume --version**

7. Flume Examples

Send local regular file *myStuff/foo.txt* to remote directory */home/auser*:

```
flume myStuff/foo.txt auser@adomain.com:/home/auser
```

Results in regular file */home/auser/foo.txt*.

Send local regular file *myStuff/foo.txt* to remote regular file named */home/auser/blah.txt*:

```
flume myStuff/foo.txt auser@adomain.com:/home/auser/blah.txt
```

If */home/auser/blah.txt* did not exist, it is created by copying the source file with the new name. If */home/auser/blah.txt* did exist as a regular file, it is overwritten.

Send the directory *myStuff/movies* and all its contents to remote directory */home/auser*:

```
flume –r myStuff/movies auser@adomain.com:/home/auser
```

results in directory */home/auser/movies* containing all the contents of the source directory.

Send the directory *myStuff/movies* and all its contents to remote directory named */home/auser/hereTheyAre*:

```
flume –r myStuff/movies auser@adomain.com:/home/auser/hereTheyAre
```

Assuming */home/auser/hereTheyAre* did not exist, results in directory */home/auser/hereTheyAre* containing all the contents of the source directory (i.e., *myStuff/movies* is recursively copied but the top directory is renamed *hereTheyAre)*. However, if */home/auser/hereTheyAre* was already a directory, the result would be */home/auser/hereTheyAre/movies* and all of its contents.

Send regular file *myProject/spec.txt* to remote regular file */home/auser/myProject/spec.txt* only if the local source file is newer than the remote destination file:

```
flume –u myProject/spec.txt
auser@adomain.com:/home/auser/myProject
```

Send any regular file under local directory *myProject* to the remote directory */home/auser/myProject* only if the local file is newer than its corresponding remote version or if the remote version does not exist:

```
flume –r –u myProject auser@adomain.com:/home/auser/myProject
```

Retrieve remote regular file */home/auser/foo.txt* to local directory */home/me*:

```
flume auser@adomain.com:/home/auser/foo.txt /home/me
```

results in regular file */home/me/foo.txt*.

Retrieve remote directory */home/auser/myData* and all its contents to local directory */home/me*:

```
flume –r auser@adomain.com:/home/auser/myData /home/me
```

results in local directory */home/me/myData*.

Retrieve any regular file under remote directory */home/auser/myData* that is newer than its corresponding local version under directory */home/me/myData*:

```
flume –r –u auser@adomain.com:/home/auser/myData /home/me
```

Use tar at both the sending and receiving ends with the data piped through ***Flume***.

Sending from local to remote:

```
tar -cvf - /home/auser/myData |\
flume -W -v - auser@adomain.com:- \
--remoteCmd "(cd /new/remote/location ; tar -xvf -)"
```

Retrieving from remote to local:

```
flume -W -v auser@adomain.com:- -\
--remoteCmd "(cd /location/to/get ; tar -cvf - .)" |\
(cd /new/local/location ; tar -xvf -)
```

8.Other Options

8.1 PAM Configuration –

NOTE: As of build 2.2.2.1105, ***Flume*** ships with PAM disabled. If needed a version with PAM enabled can be supplied.

Flume can use Pluggable Authentication Modules (PAM) to allow additional control in authenticating the Flume client.

See http://www.kernel.org/pub/linux/libs/pam/whatispam.html and http://en.wikipedia.org/wiki/Pluggable_Authentication_Modules for additional information about PAM. PAM authentication mechanisms for the Flume client ('flume') are specified in /etc/pam.d/flume. As part of the standard installation we supply a file that simply turns off the authentication but has commented examples of some other possibilities.

8.2 SLES 9 with either 32-bit and 64-bit kernels Version

At this time, the regular Linux 2.6 kernel release of ***Flume*** is known to not work on SUSE Pro 9 or SUSE Linux Enterprise Server (SLES) 9 but a modified version is available that works with SLES 9 with either 32-bit and 64-bit kernels.

8.3 Other Operating Systems

There are other operating systems where the code has successfully run. Please contact Saratoga Data Systems for more information.

Appendix A - System Administration Information

Obtain Two (or More) Linux Systems for Use in Testing

Hardware

- Intel or AMD x86 or x86_64 (amd64 or Intel64)
- Physical machine or virtual machine (e.g. hosted in VMware)
- Prefer 4 cpu/core systems but dual or single also work
- Sufficient RAM for OS (typically >256 MB)
- Sufficient disk for OS, Flume installation (~5 MB), and test files
- Disks may be network mounts including NFS or Samba/CIFS

Operating System with Linux 2.6 kernel

- Red Hat Enterprise Linux (RHEL) or CentOS 4 or 5, 32-bit or 64-bit
- SUSE Linux Enterprise Server (SLES) 10, 32-bit or 64-bit, SLES 9 64-bit

Verify Software Packages and Configuration

OS Software Required Packages

- **Text Editor** (vi OK, emacs better)

```
[root@q-centos5x64 ~]# which vi
/bin/vi
[root@q-centos5x64 ~]# rpm -qf /bin/vi
vim-minimal-7.0.109-6.el5

[root@q-centos5x64 ~]# which emacs
/usr/bin/emacs
[root@q-centos5x64 ~]# rpm -qf /usr/bin/emacs
emacs-nox-21.4-20.el5
emacs-21.4-20.el5
```

- **SSH**: openssh, openssh-clients, openssh-server, openssh-askpass

```
[root@q-centos5x64 ~]# rpm -qa | grep ssh
openssh-4.3p2-36.el5_4.2
openssh-clients-4.3p2-36.el5_4.2
openssh-askpass-4.3p2-36.el5_4.2
```

SSH Configuration Check

- Verify **ports** used by ssh and sshd (standard is 22)

```
[root@q-centos5x64 ~]# grep ^Port /etc/ssh/ssh*_config
/etc/ssh/ssh_config:Port 22
/etc/ssh/sshd_config:Port 22
```

- verify that ssh server starts at boot (turn on if necessary):

```
[root@q-centos5x64 ~]# chkconfig --list | grep ssh
```

```
sshd            0:off   1:off   2:off   3:off   4:off   5:off   6
:off
[root@q-centos5x64 ~]# chkconfig --levels 2345 sshd on
[root@q-centos5x64 ~]# chkconfig --list | grep ssh
sshd            0:off   1:off   2:on    3:on    4:on    5:on    6
:off
```

- sshd server must be running:

```
[root@q-centos5x64 ~]# service sshd start
Starting sshd: [  OK  ]
[root@q-centos5x64 ~]# service sshd status
openssh-daemon (pid  2377) is running...
```

Firewall Configuration Check

- determine if firewall (**iptables**) is running

```
[root@q-centos5x64 ~]# service iptables status
Table: filter
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    fail2ban-SSH  tcp  --
  0.0.0.0/0           0.0.0.0/0           tcp dpt:22
... lots of messages if firewall is running...
```

- simplest option - turn off firewall

```
[root@q-centos5x64 ~]# service iptables stop
Saving firewall rules to /etc/sysconfig/iptables: [  OK  ]
Flushing firewall rules: [  OK  ]
Setting chains to policy ACCEPT: filter mangle nat [  OK  ]
Unloading iptables modules: [  OK  ]
[root@q-centos5x64 ~]# service iptables status
Firewall is stopped.
```

- alternative - open firewall ports needed by Flume (specified in /etc/flume/ports)
  Details are beyond the scope of this document!

Use System Names to Enhance Convenience and Clarity

Set Hostname

- RATIONALE: a meaningful hostname (as opposed to "localhost") reduces confusion at the command line and if the console is logged for later inspection.
- RECOMMENDED: match hostname and virtual machine name in VMware (e.g. FLUME_1)
- Most easily done with "Network" GUI on DNS tab
- Reboot if necessary
- Hostname will show in command line prompt to the right of @

Use /etc/hosts to make systems easier to find

- RATIONALE: simple and descriptive names for each system reduce confusion at the command line and in any console logs.
- ALTERNATIVE: have DNS configured with this information
- Add lines with IP and names(s) for all systems used in testing: (e.g. flume1, flume2, flume3)
- Include hostname version per above (e.g. FLUME_1)
- Include simpler version (e.g. all lowercase flume1 and/or really short f1)
- Include version with fully qualified domain name (e.g. flume1.mycompany.com) to speedup boot time.
- Example:

```
[root@flume1 ~]# head /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1    localhost.localdomain localhost
10.34.49.21  flume1.company.com flume1 FLUME_1
10.34.49.22  flume2 FLUME_2
10.34.49.23  flume3 FLUME_3
```

Use public / private key pair to allow unprompted ssh (and flume) connections

- RATIONALE: without this there will be a password prompt every time a connection is made with ssh, scp, or flume. This gets old in a hurry!
- generate key pair with ssh-keygen (recommended type dsa)

```
ssh-keygen -t dsa

Hit Enter when prompted for passphrase (no passphrase)
creates .ssh/, .ssh/id_dsa, and .ssh/id_dsa.pub
```

- rename public key from id_dsa.pub to authorized_keys (or append to authorized_keys if already exists)

```
cd .ssh
mv id_dsa.pub authorized_keys

(or if file already exists
cat id_dsa.pub >>authorized_keys
rm  id_dsa.pub
)

 This allows connections back from machine2 to be without a prompt.
```

- create tar files of .ssh and copy to other systems with scp

```
cd ~
tar -cvf .ssh my-dot-ssh.tar
scp my-dot.ssh.tar machine2:
```

- ssh to other systems and unpack tar file

```
ssh machine2
tar -xvf my-dot-ssh.tar

creates .ssh/* identical to machine1
```

- make sure permissions are appropriate on both systems

```
default permissions when .ssh is created for the first
time in first step 1. seem to work.

if not, this seems to work:

cd ~
chmod 700 .ssh
chmod 400 .ssh/*
chmod 600 .ssh/known_hosts
```

- test both directions for no password prompt (you will get a prompt *on the first connection only* to accept the other hostname signature - be sure to answer yes when prompted)

```
machine1: ssh machine2
machine2:

machine2: ssh machine1
machine1
```

After all of the above is in place... Install Flume per the README file included with the distribution!

And... if there are failed runs, clean up the environment afterwards:

- RATIONALE: especially when first trying to use Flume, blocked ports or other configuration problems can result in orphaned flumed processes at the server even after killing the flume client. This can clog up the system and confuse subsequent Flume runs.
- Cleanup consists of:
  - killing any running flume or flumed processes - at both the client (less likely) and the server system
  - cleaning out the /var/run/flume/ports file to re-enable the use of ports used in earlier runs
  - waiting long enough after killing processes for ports used by them to no longer be held. This can be checked with the netstat -a -n command.
- Saratoga Data Systems, Inc. can supply a shell script (kill-fDandPorts.sh) and companion Perl program (kill-by-name.pl) to help automate this process.

Appendix B: Installation Script

A very simple installation script to perform most of the installation steps is supplied with ***Flume***. It should be modified as needed for your particular installation.

NOTE: Starting with release 2.2.3.2727, flume is also shipped as an RPM file. When this is used the script below is not used.

Flume is highly configurable with respect to file and directory locations. This includes both those used to control flume such as the ports file and those used at run time for port tracking, check points, and log files.

Note that the following script is based on standard locations as supplied in the default configuration file. If different locations are used, those locations specified in the configuration file must agree with those in the installation script.

```
#!/bin/bash
echo $0 STARTING...
FLUME_BIN_INSTALL_DIR=/usr/local/bin
echo "========================================================================="
echo Installing from directory...
pwd
echo FLUME binaries will be installed into ${FLUME_BIN_INSTALL_DIR}
echo "========================================================================="
echo UNPACK FLUME
# There should only be one flume_2.3?????????.tar.gz  present
tardone=0;
for t in flume_2.3?????????.tar.gz ;
  do if [[ $tardone == 0 ]] ;
     then echo EXPANDING TAR FILE: $t;
          tar -zxf $t;
          tardone=1;
     else echo ERROR: MORE THAN ONE TAR FILE: $t;
          echo "-----------------------------------------------";
          ls -l flume_2.3?????????.tar.gz;
          echo "-----------------------------------------------";
          echo ERROR: ABORTING installation. MORE THAN ONE tar.gz file.;
          echo "";
          exit -1;
     fi;
done;
echo "========================================================================="
echo CONFIRM successful unpack...
if [[ -x bin/ARCH.linux_26_i86/release/flume ]];
    then echo FLUME binary found. GOOD!;
    else echo ERROR: untar failed and flume binary is not present.;
         echo ERROR: ABORTING installation.;
         echo "";
         exit -1;
fi;
echo "========================================================================="
echo SETUP /etc/flume with default files...
mkdir -p /etc/flume
chmod -v 777 /etc/flume
cp -v src/doc/config /etc/flume/
chmod -v 666 /etc/flume/config
mkdir -p /etc/flume/license
chmod -v 777 /etc/flume/license
cp -v src/doc/typical-ports /etc/flume/ports
chmod -v 666 /etc/flume/ports
cp -v *flume*.lic /etc/flume/license
chmod -v 444 /etc/flume/license/*flume*.lic
echo "-----------------------------------------------"
echo Flume PORTS file /etc/flume/ports contains...
cat /etc/flume/ports
echo "-----------------------------------------------"
```

```
echo Flume config file is in /etc/flume/config...
ls -l /etc/flume /config
echo "========================================================================="
echo SETUP /var/run/flume with default files...
mkdir -v -p /var/run/flume
chmod -v 777 /var/run/flume
# ---------------------------------------------------------------------------
touch /var/run/flume/ports
chmod -v 666 /var/run/flume/ports
# ---------------------------------------------------------------------------
mkdir -v -p /var/run/flume/check_points
chmod -v 777 /var/run/flume/check_points
echo "========================================================================="
echo SETUP /var/log/flume directory...
mkdir -v -p /var/log/flume
chmod -v 777 /var/log/flume
echo "========================================================================="
echo SETUP default PAM authentication control...
cp -v src/doc/etc-pam_d-flume /etc/pam.d/flume
chmod 666 /etc/pam.d/flume
echo "========================================================================="
echo SETUP default INETD summoning control...
echo "-------------------------------------------------------------------------"
cp -v src/doc/etc-xinetd_d-flume /etc/xinetd.d/flume
chmod 666 /etc/xinetd.d/flume
echo ""
echo "========================================================================="
echo Change to flume binary directory...
pushd bin/ARCH.linux_26_i86/release/
# Flume 2.3 outputs all messages to stderr instead of stdout
./flume --versionString >& /tmp/flume-versionString.tmp
FLUME_VERSION_STRING=`cat /tmp/flume-versionString.tmp`
rm -f /tmp/flume-versionString.tmp
FLUME_LONG_NAME=flume-${FLUME_VERSION_STRING}
echo "-------------------------------------------------------------"
echo  Copy flume version ${FLUME_VERSION_STRING}...
cp -v flume ${FLUME_BIN_INSTALL_DIR}/${FLUME_LONG_NAME}
echo "-------------------------------------------------------------"
echo  Create hard links with all executable names used by flume...
for f in * ; \
    do echo --- ; \
    rm -vf ${FLUME_BIN_INSTALL_DIR}/$f; \
    ln -v ${FLUME_BIN_INSTALL_DIR}/${FLUME_LONG_NAME} ${FLUME_BIN_INSTALL_DIR}/$f; \
done;
echo "-------------------------------------------------------------"
echo Last eight flume entries in ${FLUME_BIN_INSTALL_DIR} are ...
ls -lrt ${FLUME_BIN_INSTALL_DIR}/flume* | tail -8
echo "-------------------------------------------------------------"
echo Return to installation directory...
popd
pwd
echo "========================================================================="
echo INSTALLED FLUME SHOULD BE ${FLUME_LONG_NAME} with version message:
${FLUME_BIN_INSTALL_DIR}/flume --version
echo "-------------------------------------------------------------------------"
echo INSTALLED FLUME IS:
which flume
flume --version
echo "-------------------------------------------------------------------------"
echo INETD TEST: Checking /etc/services for flume entry...
grep flume /etc/services
echo NOTE: /etc/services must contain a flume entry for INETD summoning to work - see
Users Manual.
echo "-------------------------------------------------------------------------"
echo You MUST configure flume before operation!
echo ""
echo You can reliably configure flume with the program 'flumec'
echo "-------------------------------------------------------------------------"
echo $0 DONE.
```

Appendix C: Flume Return Codes

In order to facilitate scripting of flume transfers, the ***Flume*** client has an extensive list of return codes. All negative values indicate an error that resulted in a failed transfer. A value of 0 indicates a completely normal return with no errors or warnings. All positive values represent warnings but are cases when at least one file transfer succeeded. Larger magnitudes of the return value indicate "stronger" messages.

```
unknownFailure                  = -99,
addressContinuallyNotAvailable  = -45,
failCommunication               = -44,
interrupted                     = -43,
killed                          = -42,
aborted                         = -41,
unexpectedException             = -40,
terminateException              = -39,
failThread                      = -38,
abortFileTransfer               = -37,
failCreatePipe                  = -36,
failClosePipe                   = -35,
failOpenFileDescriptor          = -34,
failForkChild                   = -33,
failAllocateMemory              = -32,
failReadInteger                 = -31,
invalidReconnectRequest         = -30,
failAssignPortNumbers           = -29,
failLoadConfigParams            = -28,
noLicense                       = -27,
licenseInitFailure              = -26,
incorrectUsage                  = -25,
failCreateLogDir                = -24,
failCreateLogFile               = -23,
failCreateRunDir                = -22,
invalidTempDir                  = -21,
failInitRunDir                  = -20,
failInitNetworkComm             = -19,
failConnectPeer                 = -18,
failSummonPeer                  = -17,
failTransmitRemoteFileList      = -16,
signatureFailure                = -15,
failTransmitSignatures          = -14,
deltasFailure                   = -13,
mergeFailure                    = -12,
failTransmitDeltas              = -11,
failSetUserId                   = -10,
userNameNotFound                =  - 9,
failOpenCheckPointFile          =  - 8,
lockCheckPointFile              =  - 7,
invalidCheckPointFile           =  - 6,
invalidCipherSpec               =  - 5,
invalidLocalPath                =  - 4,
invalidRemotePath               =  - 3,
mismatchLocalAndRemotePaths     =  - 2,
assertion                       =  - 1,
normal                          =    0,
recoverPublicKeyDecode          =    6,
jumpClock                       =    9,
```

```
failRemoveFile                  =  12,
failUname                       =  15,
suspectCheckPointFile           =  18,
ignoreInvalidOption             =  21,
invalidInternalState            =  24,
nonDefaultConfigLocation        =  27,
ignoreUndefinedParameter        =  30,
invalidParameterValue           =  33,
failReadConfigFile              =  36,
failGetLoginInfo                =  39,
failGetGroupInfo                =  42,
failReleaseSummonPort           =  45,
defaultPorts                    =  48,
recoverSocket                   =  51,
connectionTooFast               =  54,
stallCommunication              =  57,
invalidAddress                  =  60,
addressNotAvailable             =  63,
refuseConnection                =  66,
eagainConnection                =  69,
eagainThread                    =  70,
resetConnection                 =  72,
recoverConnection               =  75,
lostUDPConnection               =  78,
invalidNAK                      =  81,
fullDisk                        =  84,
missingWritePermission          =  87,
ambiguousPipeFile               =  90,
pipeNoChildProcess              =  91,
sigPipeReceived                 =  92,
mismatchSourceTargetTypes       =  93,
destinationPostDatesSource      =  96,
invalidSourceType               =  99,
tooManySymbolicLinks            = 102,
failPreserveFileAttributes      = 105,
failStat                        = 108,
failCreateSymbolicLink          = 111,
failMakeDirectory               = 114,
failOpenFileForSend             = 117,
failOpenFileForReceive          = 120,
failSha1sum                     = 123
```

Appendix D: Open Source Copyright Notices

Flume incorporates a number of free or open source software packages. Notices required by the developers follow.

librsync file differencing

Flume incorporates the **librsync** technology from
http://sourceforge.net/projects/librsync/

Copyright (C) 2000, 2001 by Martin Pool <mbp@samba.org>
Copyright (C) 2003 by Donovan Baarda abo@minkirri.apana.org.au

zlib compression

Flume incorporates **zlib** compression technology from http://www.zlib.net/
Acknowledgments:

The deflate format used by zlib was defined by Phil Katz. The deflate and zlib specifications were written by L. Peter Deutsch. Thanks to all the people who reported problems and suggested various improvements in zlib; they are too numerous to cite here.

Copyright notice:

(C) 1995-2004 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly jloup@gzip.org
Mark Adler madler@alumni.caltech.edu

Botan encryption

Flume uses the **Botan** library at http://botan.randombit.net/
Botan (http://botan.randombit.net/) is distributed under these terms:

Index